KIM-1/6502 USER NOTES

JANUARY 1977 ISSUE 3

ATTENTION!! ATTENTION!!

We've moved - here's the new address for all correspondence and subscriptions:

KIM-1 USER NOTES
c/o Eric C. Rehnke
425 Meadow Lane
Seven Hills, Ohio 44131 Phone - 216-524-7241

KIM-1 USER NOTES is published every 5 to 8 weeks. The subscription rate for U.S. and Canadian subscribers is $5.00 for issues 1 thru 6 including 1st class postage. Foreign subscriptions - $8.00 icluding 1st class air mail postage.

Payment should be made in U.S. funds with a check or money order (no cash or purchase orders) please.

To alleviate possible typographical errors, please try to submit articles in original type, single spaced on white bond so that we may cut and paste instead of re-typing. Also, if you expect a personal response to correspondence, please include a self addressed stamped envelope, to help defray expenses.

•••••••••••

Here is a correction for issue #2. - Stan Ockers informed me that page 10 address 03AF should be A9 (not 49).

•••••••••••

Hope you all had a ha ppy holiday and that your happy new year will last through 1977!

•••••••••••

KIM sure had a happy new year! Two national amatuer radio mags had application articles on KIM in their Jan. '77 issues! 73 Magazine showed how KIM could be set up to be a Morse Code Keyboard. (9-1000 WPM) with the addition of an ASCII keyboard and a few discrete components.

I had my morse keyboard up and running the very next day with a Sanders 720 keyboard and the reed relay and transistor driver from what I had laying around the shack. Boy, what an exciting feeling to just press typewriter keys and have perfect CW going out on the air! I really enjoy rag chewing (ham lingo for chewing the fat) and the Morse keyboard makes it almost effortless. The first fellow I had a QSO (contact) with had to listen to about 10 solid minutes of CW (morse code) because once I got started hacking away at the keys, I just couldn't stop! The great advantage of the software approach to implementing a morse-code keyboard really becomes apparent when you want to add fuctions or otherwise change its operation.

One of our members, Carl M. Robbins W8JDH, wrote the article on controlling a 2-meter repeater with KIM which appeared in the Jan '77 issue of QST Magazine. Complete schematics on KIM interfaces and full software listings are provided, which make this seem like the best way to go if you are setting up a repeater and want to go the microprocessor route!

Got a letter from Eliot Friedman K8OMR, 1175 Wendy Rd., Ann Arbor, Michigan 48103. Eliot is interested in contacting other radio amateur KIM owners and perhaps forming a net where information and programs could be exchanged.

•••••••••••

Some further comments on the Byte Shop #2 Tiny Basic tapes from Bob Grater: These cassette tapes are set up to run in KIM with 4K of memory added from 0400-13FF. If you want Tiny Basic for the area above the KIM Monitor (address 2000 on up) then you will have to get the paper tape version which costs $7.50 and includes the manual. The cassette version runs $9.50. Include $1.00 shipping on either of these items.

••••••••••

FROM THE FACTORY: Problems with some 6502 chips.

I have been informed of two possible problem areas with the 6502 CPU. Some of the early CPU's didn't set the zero flag correctly after register to register transfers,(TAY, TAX,TYA etc This would make it impossible to run Tiny Basic. All 6502 CPU's could have problems setting the zero flag after decimal addition when the answer equalled zero. Test your CPU with a simple program to see if you have these problems. The decimal addition problem can be gotten around by an "OR" immediate with "00" after the operation which would then set the zero flag correctly. The only method of solving the register transfer not setting zero flag would be to install a new CPU. (You would then pick up the "ROR" rotate right instruction in the process).

Copying KIM cassette tapes:

If you've tried copying KIM tapes from cassette to cassette then you already know that the copies will not be read correctly by KIM. The fix? Simply tie a .02 uf capacitor to ground from the junction of R6 and R34. Now make the new master tape. Copies from this new master should be read correctly by KIM.

KIM-2 Memory expansion:

If you wish your KIM-2 (4K) memory module to reside in the 0400-13FF position already decoded by KIM, then read on; Wire - or the KIM K1, K2, K3, K4 decoder outputs together, add a 3.3K pull-up resistor to +5 volts and tie these four lines to the address line 12 input on the KIM-2, set all the on-board DIP switches to the "off" position, and you're all set!

****

INTRODUCTION TO KIM-1 ARCHITECTURE

J. Butterfield
14 Brooklyn Avenue
Toronto M4M 2X5
Canada

This is intended to be a beginner's guide to the way KIM-1 is put together. It's mostly a hardware description with (hopefully) explanatory notes.

Because every KIM-1 owner has three fat manuals on his system, complete with detailed drawings, I'll try to save space here by referring to these manuals wherever possible.

The Address and Data Busses.

Let's start with page 24 of the KIM-1 User Manual (KIM-1 Block Diagram, Figure 3.1). We're going to concentrate on those two pipes on the left: the Address and Data Busses.

Every microsecond, the 6502 microprocessor sends out an address over the sixteen lines of the address bus. Sixteen lines are enough to address any of the 65,536 memory locations that could be fitted to KIM-1 (you only have 3,328 active addresses in the basic unit). In addition, there are a couple of other lines that accompany this bus: a Read/Write line (R/W) to tell whether the microprocessor wants to read or write memory; and a timing line, ∅2 (phi, pronounced fy, two; it's confusing on the diagram because the phi looks like a zero).

The address bus goes to all memories. The idea is that when an address is generated, one memory only (whether RAM, ROM, I/O or Timer) suddenly says, 'that's me!' and connects to the data bus. If the R/W line says, 'read', the memory concerned places its data onto the bus; if 'write', the memory takes the data from the bus (placed there by the microprocessor) and stores it. For every address, only one memory unit responds; the rest stay silent.

This points out a fundamental difference between the address and data busses. The address bus goes one way only: from the processor to the memory. But the data bus information flows both ways.

This calls for a special kind of circuitry to connect to the data bus, called 'tri-state'. Every device on the bus might be (1) sending; (2) receiving; or, (3) ignoring the bus. (That isn't exactly how tri-state is defined, but it's a good way to remember it).

LOOKING FOR PROMS???? I just picked up four 82S129 bipolar 256x4 promsto make SUPERTAPE and other such used software a permanent part of KIM. S.D. SALES CO., P.O. BOX 28810, DALLAS, TEXAS, 75228, also has 5204 EPROMS (512x8, 1 usec.) for $7.95, 1702A (1 usec. 256x8) for $6.95,8T97B's for $1.25, etc. etc. All their parts are guarenteed and they deliver in about a week. Definitely good people! Get their catalog! If you order from this, include 5% for shipping, $0.75 for orders under $10.00, and tax if you're in Texas.

—the editor—

Since a(3s bus signals go one way only, it doesn't need to tri-state circuitry, which simplifies things. However, it could cause you minor problems if you wanted to expand your KIM-1 system so as to have two processors sharing the same memory, or if you wanted to drive a TV display directly from memory. In cases like that, the problem is called DMA (Direct Memory Access) and hints on how to solve it can be found on page 112 of the Hardware Manual.

Turning to page 25 of the KIM-1 User Manual, we can see the Address and Data busses in a little more detail. A little closer examination turns up a problem: the whole address bus doesn't connect to the memory modules. In fact, only 10 bits (numbered AB0 to AB9) out of the 16 are connected. That's only enough to define 1,024 addresses. How can it work?

Well, buried in the control logic block at the upper right is an 'address decoder'. Although figure 3.2 doesn't show the connections, this circuit is looking at address bus lines AB10, AB11, and AB12. Depending on what it sees in this part of the address, it may fire up either of the ROMs (via leads K6 and K7), or the Timer/IO/RAM section of the 6530's (via lead K5), or the 1K RAM (via lead K0). Each memory unit will work only if its respective K lead is energized. So now, each unit is responding to 10+3 or 13 bits of address information.

The address decoder, incidentally, can be seen in more detail on the next page, Figure 3.3, Control and Timing. It's block U4; you can see the three lines of the address bus coming in from the left, and the select signals (K0 through K7) going out at the bottom. Incidentally, outputs K1 through K4 are not used on the basic KIM-1; but they are available to you for expansion purposes. Each lead is capable on controlling 1K (1,024) bytes of memory.

What about the other three bits of the address bus, AB13, AB14, and AB15? KIM-1 ignores them. They are not needed for the basic system, so they're not used.

This leads to an odd feature of the basic KIM-1 system. Since the three high-order bits of the address are ignored, the memory starts repeating itself after reaching 1FFF hex. For example, try storing something in address, say, 0123; then look at the contents of 2123, and 4123, etc. They're all the same location!

This explains something that can be quite troublesome to the beginner. Both the Hardware Manual (page 40) and the Programming Manual (chapter 9, pages 124-146) state that the interrupt and reset vectors are located at hexadecimal addresses FFFA to FFFF. Yet reading the KIM-1 Monitor program listings shows them to be at 1FFA to 1FFF. Very puzzling--until you realize that in the basic KIM, they are the same locations!

It also outlines something you'll need to take into account when you add memory. If you use the 4K decoded for expansion with lines K1 to K4, no special reconfiguration is necessary ... you're staying in the area where the 'duplicated' addresses won't bother you. But to go to addresses at hex 2000 and higher, read Chapter 6 of the KIM-1 User Manual very carefully.

<u>Interrupts, Reset, and the SST</u>

The 6502 has three pins which force it to branch to a given location. They are called RST (reset), NMI (non-maskable interrupt), and IRQ (interrupt request). They all differ in detail, but the manner in which they cause a branch or jump in program execution is similar; we'll discuss them together.

First: when the appropriate input pin is 'kicked', the microprocessor hardware, after doing a couple of odd jobs, digs out the address that it will jump to from a fixed location. This location varies depending on which pin was activated, but it will be in the range FFFA to FFFF hexadecimal (see page 146 of the Programming Manual).

Next step: because of the way the KIM-1 addressing is wired, you may recall that address FFFA is the same as address 1FFA. This is part of ROM (the 6530-002 chip), so that the contents of these vectors are fixed. They point at programs to handle each activity.

Let's summarize what we have so far in a table:

Pin	Address of Vector Hardware	KIM-1	Contents (Vector)	What you'll find at the vector location
NMI	FFFA-FFFB	1FFA-1FFB	1C1C	Program NMIT (Jump indirect)
RST	FFFC-FFFD	1FFC-1FFD	1C22	Prgm RST (A reset program)
IRQ	FFFE-FFFF	1FFE-1FFF	1C1F	Program IRQT (Jump indirect)

So the branches we take are completely predefined for us by the Monitor program. In the case of Reset, the system will reset the stack pointer and then proceed to the main monitor program.

But the other two are different. Programs NMIT and IRQT, to which the two respective vectors point, are nothing more than indirect jumps. And the address to which you will jump, in both cases, is stored in RAM; so you can change these addresses to make the two interrupts branch anywhere you want. (Normally, you'll want the address of monitor program SAVE, at 1C00, stored in both vectors).

So there are two kinds of vectors: the hardware vectors at 1FFA to 1FFF which you can't touch since they are in ROM; and the software vectors at 17FA to 17FF which you must set up each time you turn on the system. (Oddly enough, the software vectors include a Reset vector that isn't used).

It's good practice, every time you power up, to key in:

<u>AD</u> <u>1</u> <u>7</u> <u>F</u> <u>A</u> <u>DA</u> <u>0</u> <u>0</u> <u>+</u> <u>1</u> <u>C</u> <u>+</u> <u>0</u> <u>0</u> <u>+</u> <u>1</u> <u>C</u> <u>+</u> <u>0</u> <u>0</u> <u>+</u> <u>1</u> <u>C</u> <u>AD</u>

setting the vectors and thus enabling the ST key and SST switch.

Now let's take a little excursion into software. When you push the GO button (or hit the G key if you're lucky enough to have teletype), the Monitor does a few last things before it gives up control to your program. Refer to page 39 of the KIM User Manual (Special Memory Addresses). You'll see that locations 00F1 to 00F5 are associated with various registers of the microprocessor.

When you push the GO button, the Monitor digs out these locations and puts them in the microprocessor registers. (You can read the program that does it at locations 1DC8 to 1E87 - it's quite clever).

This means something quite important to the programmer who's doing testing. You can set the initial values of any of the registers before you run - by storing the appropriate values in F1 to F5.

Note that the Monitor does all this. As far as the microprocessor is concerned, these locations have no special status; they are no different from any other part of memory. If a program puts something in the accumulator, the contents of F3 don't change, and vice versa.

Now, we've talked about what happens when your program starts. What happens when it stops? It would be handy (for debugging and other reasons) if the registers could be dumped out to the same locations (F1 to F5).

Well, program SAVE at location 1C00 does exactly that. So if possible, we should try to terminate a program by having it exit to location SAVE. Here are several ways of doing it:

1. Finish your program with the instruction JSR SAVE1 (20 05 1C). The accumulator will be lost, but the other registers will be saved correctly.
2. Finish your program with a BRK (00) instruction, and be sure your software vector at 17FE-17FF set to the address of SAVE (1C00).
3. To stop your program while it is running, press the ST button; but be sure you have previously set the software vector at 17FA-17FB to the address of SAVE

4. If you have t the SST switch to <u>on</u>, the program will stop a one instruction -- provided you have set the sof re vector at 17FA-17FB to address SAVE. More about how this happens soon.

How does the SST switch do that, anyway? Let's return to the hardware.

The diagram on page 26 of the KIM User Manual (Control and Timing) contains the basic information, especially in the area of U26 (a NAND gate).

We can see that the SST switch, when operated, will kick the NMI interrupt pin when the following two conditions are satisfied:

1) Lead K7 is not energized. In other words, the address bus is not in the range 1C00 to 1FFF;

2) The SYNC signal is high, which means the processor is fetching an instruction. (See page 44 of the Hardware Manual).

Put these together, and what do they mean? Just this: that if the SST switch is on, and we fetch an instruction that is <u>not</u> in the Monitor, the NMI interrupt will be signalled. The instruction in progress will be completed. Then, if we have set up our vector correctly, the micro will return to the executive, neatly dumping the registers as it does so. (Pressing GO for the next instruction reloads the dumped values so that the next step continues with exactly the conditions left by the previous one).

This is a really neat mating of hardware with software. The hardware allows the Monitor to run freely; but any other instructions are immediately interrupted, and the Monitor takes over again. The software, on the other hand, stacks away the register contents, and restores them when the GO button is pressed again. Elegant, huh?

Last word on interrupts: if you plan to expand the KIM-1 system, all of these features are available for you to modify. Chapter 6 of the KIM User Manual (Expanding Your System) gives quite a bit of material on this.

You might like to run the following note: "Tired of squinting at the 6500 Instruction summary--and mistaking B for 8, amoung other things? You might like a copy of my large type summary summary, which regroups by usage, and lists mnonics, hex code and formula for the effect of the command. For a Xerox copy, send a self-addressed stamped envelope with an extra 9¢ stamp(loose) for one copy or 13¢ stamp for 2 copies to: Mike Firth/6500
4712 Northway Dr.
Dallas, TX 75206"

Digital Tape Drives for KIM-1

Robert E. Haas
2288 Blackburn St.
Eugene, OR 97405

As I stated in a previous newsletter, I am developing an assembler for KIM, and I planned to use the KIM tape I/O system for source input and object output. However, I have encountered several problems including low reliability, slow speed, and general frustration with the limitations of audio cassette recording. I have instead decided to interface a commercial-quality digital storage device to my system. I have chosen the Mini-Raycorder Model 6409 by Raymond Engineering, Middletown, Ct. This drive uses a miniature version of the standard Phillips cassette, a digital-certified version of the one being used in some newer dictating equipment. The drives cost $350 in singles, declining to less than $200 in large quantities. I would be willing to organize a group purchase of the drives to take advantage of quantity pricing. I will distribute my interface hardware and software design, as well as my assembler to anyone participating in the group purchase, for the cost of distribution.

The hardware portion of the interface consists of 4 bits of latched output and 5 bits of input (already available on the basic KIM-1). The dirves give the computer full control over tape motion (forward/rewind), and read/write status, and let the computer sense cassette-in-place, tape position (beginning or end), cassette write-protect, and cassette side. The technical specs of the drives are:

Recording mode: bit serial, bi-phase, 800 bits/inch.
Tape speed: 3 in/sec forward, 20 in/sec rewind.
Transfer rate: 2400 bits/sec (300 bytes/sec).
Tape length: 50 feet. Capacity: 60,000 bytes/side (2 sides)

Anyone interested in the group purchase should send me a stamped business-sized self-addressed envelope, and indicate how many drives you wish to purchase.

Ronald Kush
3108 Addison Court
Cornwells Heights, Pa. 19020

BUT FIRST SOME CRITICISM...

1. The relative branch table has become one of my most useful tools once I figured out how it works. Perhaps a word of explanation was in order.

2. The driver circuit for the Kluge Harp originally presented in October 1975, BYTE, consisted of a 7437 (not mentioned in the users notes) arranged as a set-reset flip flop and buffer. The same circuit in the users's notes does nothing more than buffer and could just as well be replaced with the original driver circuit presented on Page 57 of the Kim User Manual. I am, also, not sure about the propriety of incrementing the data direction registers; as opposed to keeping PBDD & PADD as outputs and incrementing the data registers.

3. There seems to be some confusion regarding the interval timers and how to use them. Mr. Lutz listed the addresses of the "other timer" in the same manner as the Kim Program listings does on Page 3 of the User Manual. This was confusing to me, and I hope to clarify the timers' applications.

4. I should hope that the "User Notes" does not become a software catalog; where the average entry goes something like - "I just developed a program for Chinese checkers, which I'll sell to the members for just $10.00 plus postage". After reading an issue of "User Notes" I feel a tremendous obligation to write something; whether it be a program or just a letter of appreciation for the wealth of information that the notes contain. I should hope that the other members feel the same and are willing to share their software in the knowledge that the hours they spend developing a program will be repaid many fold by the cumulative contributions of many authors of our group. It's the only way that the "Users Notes" can remain viable.

INTERVAL TIMERS

USING THE INTERVAL TIMER AS AN IRQ INTERRUPT

First read paragraph 1.3.2.1 "Applications for Interrupts" in the Kim Hardware Manual. After reading the above paragraph it becomes easier to imagine the interval timer as a separate entity just like a keyboard, control panel or scanned display. The Timer is used to independently time a certain interval during which a program is operating. The "during which" is important because it allows the program to do other things besides waiting in a delay loop for the interval to time out. This independent operation (non-reliance on delay loops within the program" makes the device particularly valuable. Once the timing interval has elapsed it is necessary for the timer to signal the microprocessor.

This is done with the IRQ interrupt which enters the processor through PB7. If PB7 is set up as an input (0 in PBDD), a low on that pin causes the program to jump to whatever is in the IRQ Vector located at 17FE, 17FF. In other words, say you want to do a program in 0000 to 0100 for a short time then switch to 0300 and do a different program. By loading 0300 in the IRQ Vector and setting the timer this can be accomplished.

TIMER ADDRESSING

As can be seen in the address table, A3 controls whether or not PB7 is enabled as an interrupt line.

If you are not using the interrupt capability of the timer, read or write into 1704 to 1707. As an example, during a loop in a program you may continually check to see if the timer is done using the LDA (1704 to 1707) and the Bit instruction.

If you are using the interrupt capability, use addresses 170C to 170F. The important thing here to remember is that when reading the time after an interrupt go back to 1704 to 1707 so as not to enable PB7 again until required.

ADDRESS TABLE

ADDRESS 6530-003	ADDRESS 6530-002	LEAST SIG. ADD. BITS A3	A2	A1	A0	TIME DIVISION	USE
1704	1744	0	1	0	0	÷ 1T	Use these addresses to read or write to the timer when not using interrupt mode on PB7 or when reading after interrupt A3 = 0
1705	1745	0	1	0	1	÷ 8T	
1706	1746	0	1	1	0	÷ 64T	
1707	1747	0	1	1	1	÷ 1024T	
1708	1748	1	0	0	0	-	These addresses are not used with interval timer A2 = 0
1709	1749	1	0	0	1	-	
170A	174A	1	0	1	0	-	
170B	174B	1	0	1	1	-	
170C	174C	1	1	0	0	÷ 1T	These addresses are used when the IRQ interrupt capability at PB7 is needed A3 = 1
170D	174D	1	1	0	1	÷ 8T	
170E	174E	1	1	1	0	÷ 64T	
170F	174F	1	1	1	1	÷ 1024T	

Program to check KIM-1 keyboard condition

Cass R. Lewart
12 Georjean Dr.
Holmdel
New Jersey 07733

The following program will check individual keys for excessive bouncing. If an oxide layer builds on the contact surface depressing a key will cause several "bounces" or on and off conditions which may cause multiple or erroneous entries. The program checks the key every n microseconds and writes either FF for contact open or a different code into page 2 of memory. Thus we get 255 samples of the contact closure which we can easily review by calling the previously described Display routine (set MULT in that routine to 1 for faster scan). Set the word #10 in the keyboard program to 00 to check keys 0 - 6, to 02 to check keys 7 - D or to 04 to check the remaining keys E through PC. You can not check keys ST and RS. The time constant between successive samples is set by the value in word #23. The multiplier value is 4 microseconds times word #23.

```
00  A9 FF          LDA #FF      Set initial delay
02  8D 07 17       STA 1707
05  2C 07 17    1  BIT 1707     Check timer
08  10 FB          BPL 1
0A  A9 1E          LDA #1E      Set PB1-PB4 to output
0C  8D 43 17       STA 1743
0F  A9 xx          LDA xx       xx = 00, 02 or 04 (see text)
11  8D 42 17       STA 1742
14  A9 80          LDA #80      Set PA0 - PA6 to input
16  8D 41 17       STA 1741
19  AD 40 17    2  LDA 1740
1C  C9 FF          CMP #FF      Check for key depression
1E  F0 F9          BEQ 2
20  A2 00          LDX #00      Key has been depressed
22  A0 yy       3  LDX #yy      yy - multiplier 5 - 10
24  88          4  DEY
25  D0 FD          BNE 4
27  AD 40 17       LDA 1740     Delay finished
2A  9D 00 02       STA,X        Write on page 2
2D  E8             INX
2E  D0 F2          BNE 3        Continue with next sample or end
30  00             BRK          Stop
```

Note: The reason for not using AK or GETKEY routines in this application is that they are too slow to detect the bouncing.

To check e.g. key "A" set word 10 to 02, start running the program at 00 by pressing GO. The display goes blank until you press A. Go then to location 200 and start reviewing contents of the 2-nd page. Each FF means a contact bounce.

EDITORS NOTE: I changed the BREAK instruction to a JMP to 1C4F to get the program operational.

This program can detect the bounce of ANY switch by just hooking it up to the proper keyboard access pins on the KIM edge connector.

Cass R. Lewart
Holmdel, NJ 07733

Display Routine

This routine will display any program showing each successive location and the contents of that location. The routine is fully relocatable. By storing in the 17FA and 17FB locations the starting address of this routine one can use the ST key to start the program. The display can be stopped by pressing RS and continued by pressing ST again. The program starts displaying consecutive locations starting with the location shown in the display by pressing ST. The second program location Mult controls the display time. With value 04 it is .4 sec per location.

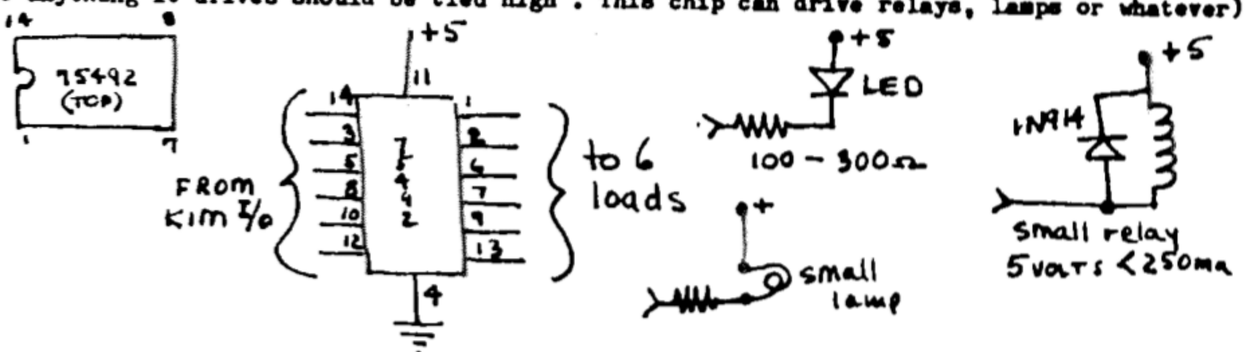
```
00   A2 04      3  LDX Mult
02   8A         2  TXA
03   48            PHA
04   A9 62         LDA #62 (.1sec/cycle)
06   8D 07 17      STA 1707  Load timer
09   20 19 1F   1  JSR SCANDS Display
0C   2C 07 17      BIT 1707 Check timer
0F   10 F8         BPL 1
11   68            PLA
12   AA            TAX
13   CA            DEX
14   D0 EC         BNE 2
16   E6 FA         INC FA
18   D0 E6         BNE 3
1A   E6 FB         INC FB
1C   D0 E2         BNE 2
```

e.g. to start displaying at 210: AD,0,2,1,0,ST
if the DISPLAY starts at 300: AD,1,7,F,A,DA,0,0,+,0,3,AD,go to desired location, ST ...

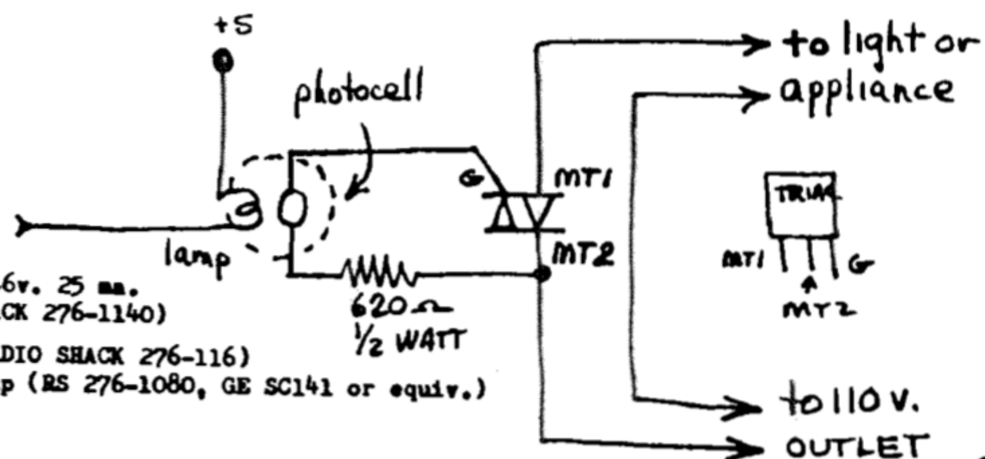
Mr. Lewart also enclosed additional details about accessing the keyboard and display which will be included in an upcoming issue — the editor

MORE FROM CASS LEWART

1. KIM LED INTERFACE: The inexpensive ($0.45 from Altaj Electronics) IC #75492 LED DRIVER provides an interface for 6 LED's or lamps (editors note- the 75492 can SINK up to 250 ma. so anything it drives should be tied high . This chip can drive relays, lamps or whatever)



2. KIM AC INTERFACE: This circuit can be used to control lights, appliances,etc, up to 600 watts. Though Sigma light-couplers are available from Allied Electronics, it is easy to make one by wrapping a lamp and a CdS photocell with dark masking tape.



low-current lamp-6v. 25 ma.
(RADIO SHACK 276-1140)

CdS photocell (RADIO SHACK 276-116)
Triac- 200v./6 Amp (RS 276-1080, GE SC141 or equiv.)

p.9

FROM: Joe Laughter
Univ. of Tennessee
Medical Units
951 Court Ave.
Rm. B23D
Memphis, Tenn., 38163

HERE'S A WAY TO TURN KIM INTO A FREQUENCY COUNTER:

FREQ. COUNT

Counts freq. using input PBØ, max rate 20 Khz; counts DATA for 1 sec. to count for 10 sec. load #29 into adr 60. Uses PB7 for int. req. (connect PB7 to INT. REG.)

```
0000   A9 01                LDA #01
0002   85 65                STA TMECNT
0004   F8                   SED
0005   A9 36                LDA # INT LOW     SET INT. VECTOR
0007   8D FE 17             STA FE
000A   A9 00                LDA # INT HIGH
000C   8D FF 17             STA 17FF
000F   58                   CLI
0010   00                   BRK
0011   EA                   NO OP
0012   AD 02 17   CK LOW    LDA PB            CHECK FOR INPUT LOW
0015   29 01                AND #01
0017   DO F9                BNE CK LOW
0019   AD 02 17   CK HIGH   LDA PB            CHECK FOR INPUT HIGH
001C   29 01                AND #01
001E   FO F9                BEQ CK HIGH
0020   18                   CLC               ADD COUNT TO TOTAL
0021   A9 01                LDA #01
0023   65 F9                ADC F9
0025   85 F9                STA F9
0027   A9 00                LDA #00
0029   65 FA                ADC FA
002B   85 FA                STA FA
002D   A9 00                LDA 00
002F   65 FB                ADC FB
0031   85 FB                STA FB
0033   4C 12 00             JMP CK LOW
0036   48         INT.      PHA               CHECK TIME
0037   A9 90                LDA #90
0039   8D 04 17             STA 1704
003C   2C 07 17             BIT 1704
003F   10 FB                BPL DELAY
0041   A9 F4                LDA #F4           SET TIMER FOR ANOTHER INT.
0043   8D OF 17             STA 17OF
```

0046	C6 65		DNC TME CNT	CHECK RE ..ING TIME
0048	FO 02		BEQ DISP	IF ZERO DISPLAY COUNTS
004A	68		PLA	
004B	40		RTI	
004C	A9 FF	DISP.	LDA FF	SET DISPLAY LOOP CNT
004E	85 66		STA SCANCT	
0050	20 1F 1F	OUT	JSR SCANDS	OUTPUT DATA
0053	C6 66		DEC SCANCT	DEC. LOOP CNT
0055	DO F9		BNE OUT	REPT. DISPLAY TILL LOOP
0057	A9 OO		LDA OO	COUNT IS ZERO
0059	85 F9		STA F9	SET TOTAL COUNTS TO ZERO
005B	85 FA		STA FA	
005D	85 FB		STA FB	
005F	A9 05		LDA 05	RESET 1 SEC. TIMER
0061	85 65		STA TMECNT	
0063	68		PLA	
0064	40		RTI	
0065	05		*DATA (TMECNT)	
0066	FF		*DATA (SCANCT)	

Don Box
1250 White Oak Dr.
Cookeville, Tn. 38501

Dear Eric:

As a fellow KIM-1 fan may I wish you well on the USER NOTES undertaking; they are most useful and interesting. As a small contribution (if anyone is interested) I am enclosing a photo of how I mounted my KIM-1 in a cabinet and provided expansion room for future projects.

One of my interest was driving outside devices and I came up with the following scheme to give up to 16 I/O ports (8-bits ea). The system uses six pins of port B and three pins of port A plus some external hardware. The amount of external hardware required depends upon the number of devices connected but runs about three TTL chips per device plus the hardware driver.

Driver software is in RAM, I used locations 1780 thru 17DB. An input or output to a device requires about 300 microsec if the device is ready to accept or receive data.

Figure 1 shows the hardware driver, figure 2 a typical device interface and figure 3 the software listing. I use an old Model 15 (5-level) teletype and a UART, this interface is shown in figure 4.

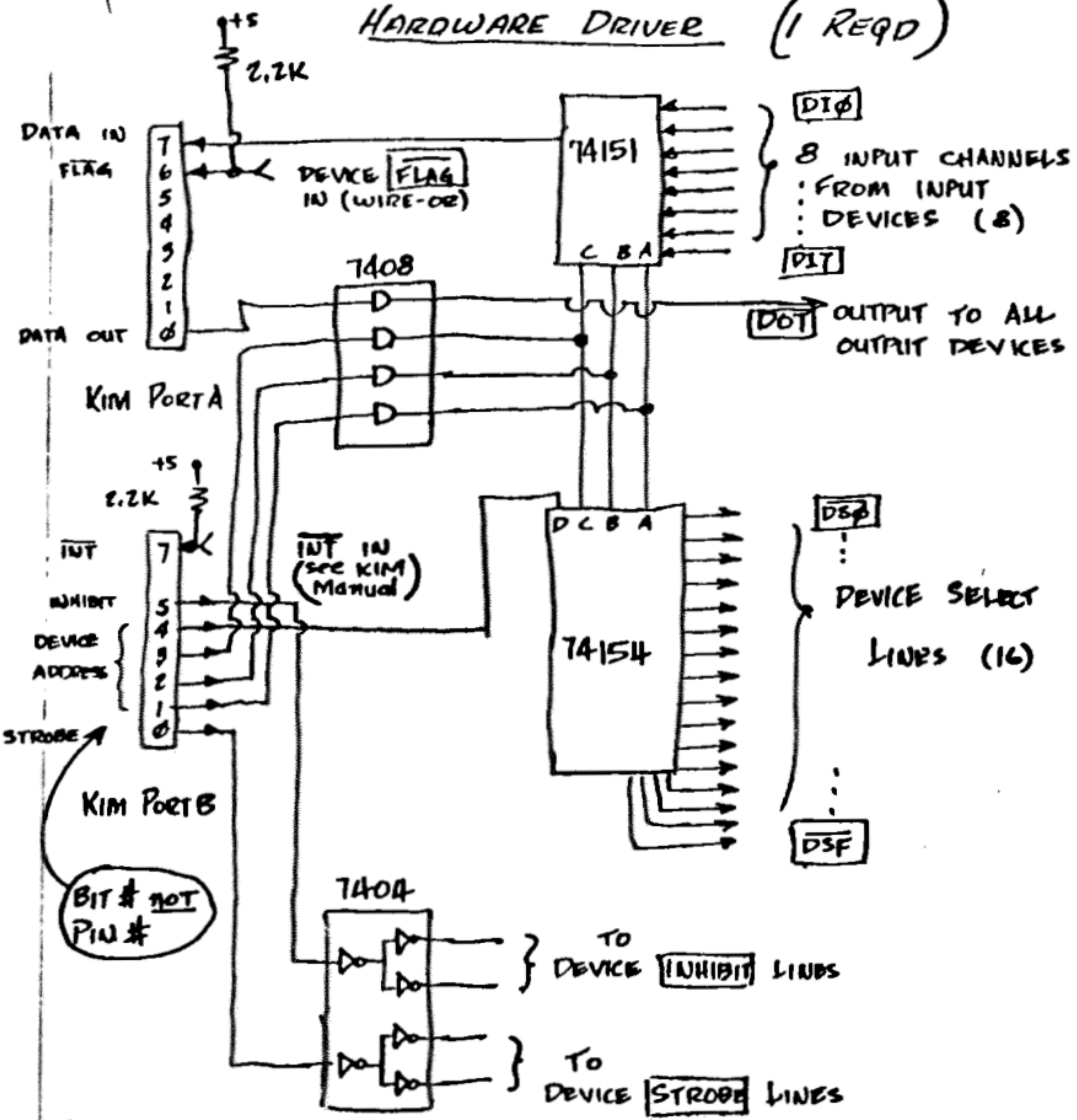
I can supply more details if anybody wants if they will send a S.A.S.E.

Keep up the good work--

Don Box

FIGURE 1

HARDWARE DRIVER (1 REQD)



NOTE:

	Location		
INITIALIZE:	17Ø1	TO	HEX 'Ø1'
	17Ø3	TO	HEX '3F'

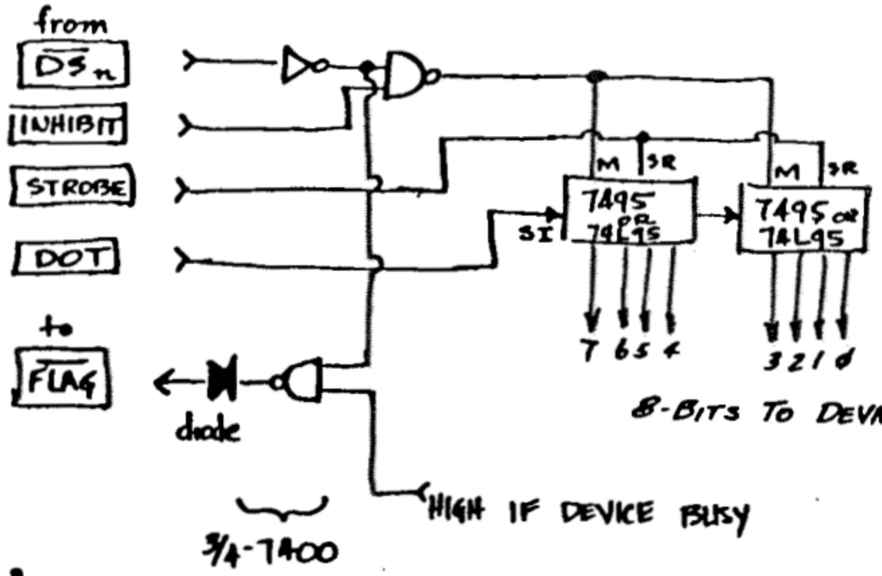
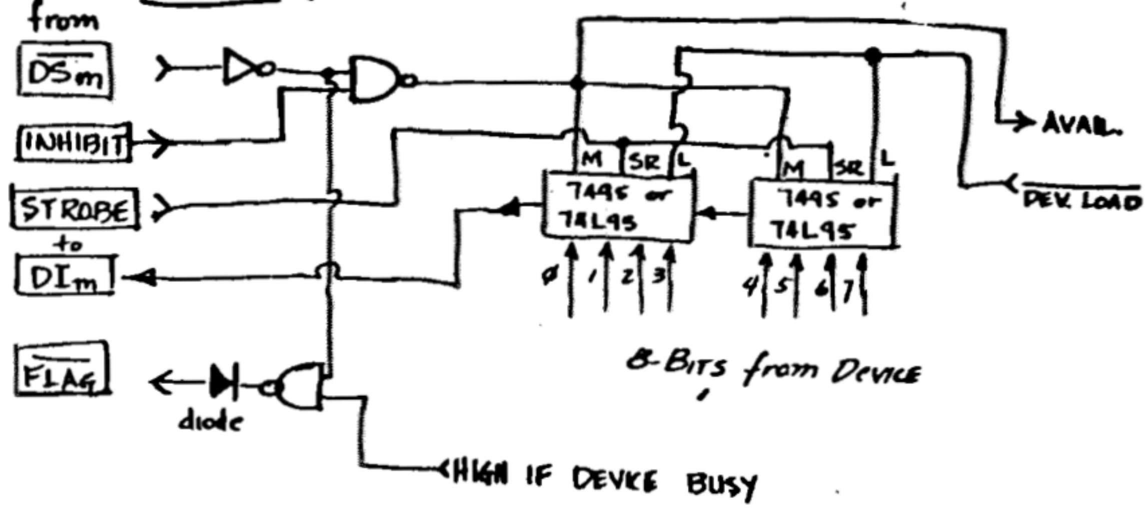


FIGURE 2
TYPICAL DEVICE INTERFACE

TYPICAL OUTPUT INTERFACE (max. of 8 possible)

FIGURE 2 (cont'd)



TYPICAL INPUT DEVICE INTERFACE (max. of 8)

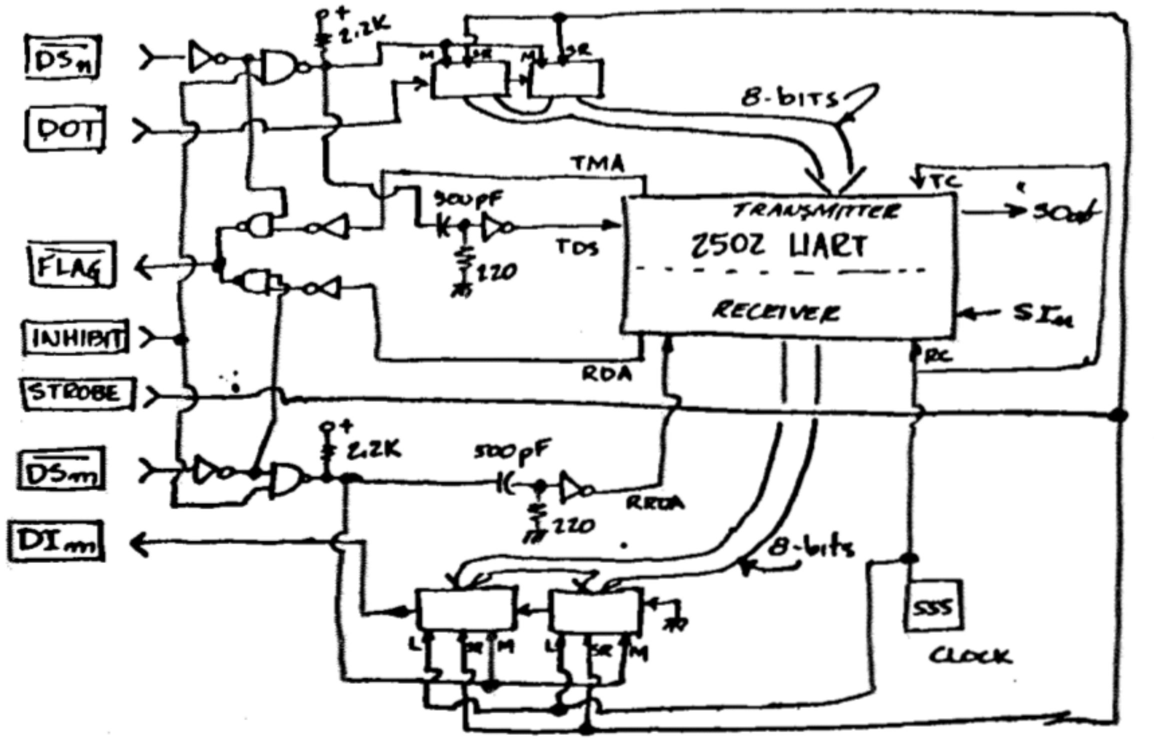
Software I/O driver

FIG 3:

```
                ;INPUT SUBROUTINE
1780  20 CA 17  GETBYTE:  JSR CKFLAG    ;SEE IF DEVICE AVAIL.
1783  50 FB               BVC GETBYTE   ;NO WAIT
1785  A0 08               LDY #$08      ;SET BIT COUNTER
1787  20 D4 17            JSR INHIBIT   ;INHIBIT DEVICE
178A  4A        RDBIT:    LSR A         ;MOVE OVER
178B  2C 00 17            BIT PAD       ;CHECK BIT
178E  30 0D               BMI AONE      ;FOR A 1
1790  EE 02 17  NEXT:     INC PBD       ;NOPE --- SHIFT
1793  CE 02 17            DEC PBD       ;VIA STROBE LINE
1796  88                  DEY           ;COUNT
1797  D0 F1               BNE RDBIT     ;DO MORE
1799  8E 02 17            STX PBD       ;ENABLE DEVICE
179C  60                  RTS           ;AND RETURN
179D  09 80     AONE:     ORA #$80      ;MAKE BIT 7 A ONE
179F  30 EF               BMI NEXT      ;AND STROBE
  :
                ;OUTPUT SUBROUTINE
17AA  20 CA 17  PUTBYTE:  JSR CKFLAG    ;SEE IF BUSY
17AD  50 FB               BVC PUTBYTE   ;YES, WAIT
17AF  A0 08               LDY #$08      ;BIT COUNTER
17B1  48                  PHA           ;SAVE BYTE
17B2  20 D4 17            JSR INHIBIT   ;INHIBIT DEVICE
17B5  68                  PLA           ;RESTORE BYTE
17B6  8D 00 17  BURP:     STA PAD       ;SET OUT BIT 0
17B9  EE 02 17            INC PBD       ;STROBE FOR
17BC  CE 02 17            DEC PBD       ;SHIFT
17BF  4A                  LSR A         ;MOVE OVER
17C0  88                  DEY           ;COUNT
17C1  D0 F3               BNE BURP      ;DO MORE
17C3  8E 02 17            STX PBD       ;NOPE, ENABLE
17C6  60                  RTS           ;AND RETURN
  :
                ;MISC SUBROUTINES
17CA  8E 02 17  CKFLAG:   STX PBD       ;SELECT DEVICE
17CD  2C 00 17            BIT PAD       ;FLAG BIT 6
17D0  60                  RTS           ;RETURN V-BIT
  :
17D4  8A        INHIBIT:  TXA           ;GET DEV. ADDR.
17D5  09 20               ORA #$20      ;SET INHIBIT BIT
17D7  8D 02 17            STA PBD       ;INHIBIT DEVICE
17DA  60                  RTS           ;RETURN
  :
```

FIGURE 4

UART INTERFACE



7404 1-REQD

7403 1-REQD

7495 or 74L95 4-REQD

2502 UART 1-REQD

555 CLOCK 1-REQD

555 TIMER
As ASTABLE
@ 16X baud rate

```
to Output a byte
            LDA  byte
A2 #$xx     LDX  DEV
20 AA 17    JSR       PUTBYTE
```

```
to input a byte
A2 #$xx     LDX  DEV
20 80 17    JSR  GETBYTE
- -         STA  byte
```

END

HAVE YOU BEEN HAVING PROBLEMS OPERATING YOUR KIM-1 CASSETTE SYSTEM RELIABLY ON AC POWER? THE MAIN PROBLEM MAY BE DUE TO AN UNREGULATED POWER SUPPLY.

SINCE ACQUIRING MY KIM-1 SYSTEM I HAVE HAD A PROBLEM OPERATING MY TWO AUDIO CASSETTE UNITS WITH AC POWER. THE PROBLEM SYMPTOM WAS NOT BEING ABLE TO ACQUIRE SYNCHRONIZATION. HOWEVER THEY OPERATED FINE ON BATTERIES. EXAMINING THE CASSETTE OUTPUTS, I FOUND A HIGH AC RIPPLE AND POOR VOLTAGE REGULATION CAUSED BY UNREGULATED POWER SUPPLIES.

MY SOLUTION WAS TO DESIGN AND BUILD SEPARATE REGULATED POWER SUPPLIES FOR EACH UNIT(6.0V AND 7.5V). THIS HAS YIELDED VERY SATISFACTORY RESULTS. SEVERAL TIMES THE UNITS HAVE READ 120-256 BYTE DATA PAGES WITHOUT ERROR.

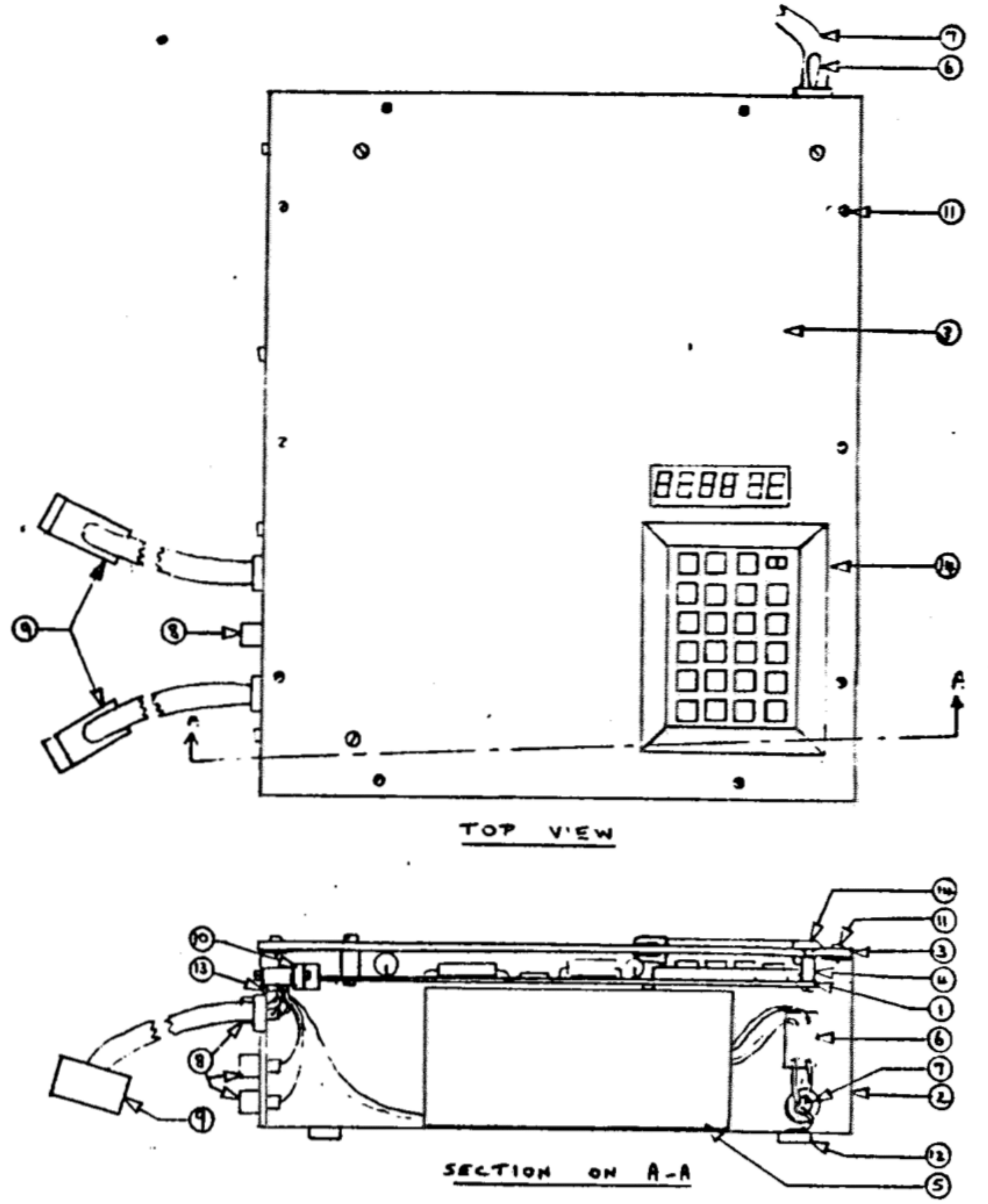
RECENTLY, I SUCCESSFULLY READ A COUPLE OF 256 BYTE PAGES USING JIM BUTTERFIELDS SUPERTAPE. I AM LOOKING FOWARD TO ITS TIME AND TAPE SAVINGS.

CHUCK BALSER
14 OVERBROOK CIRCLE
NEW HARTFORD, N.Y. 13413

HERE'S AN IDEA FOR PACKAGING KIM:

FROM: NIGEL NATHAN
12 STONEHOLM ST.,
BOSTON, MASS. 021

I used a standard Bud chassis, turned upside down, with a clear plastic top. The latter I painted black on the underside, except for the display area and the keyboard cutout. I had originally planned to project the keyboard through the opening, but had too hard a time trying to desolder it from the circuit board. The present arrangement is quite satisfactory and the circuit board has complete protection. The power supply is at one end, so there is plenty of room for expansion circuitry inside. I use wirewrap backplane sockets to bring out the applications connections as DIP plugs are male (live pins) and fragile. I use wirewrap extensively.



1) KIM-1
2) 10"x12"x3" Alum. chassis
3) 10"x12"x1/8" plexiglass
4) 7/16" long standoffs (5)
5) Power supply
6) Power switch
7) Line corD
8) Phono jacks for tape recorder (3)
9) Wirewrap backplane connectors (2) for applications
10) Applications connector
11) Sheet metal screws (10)
12) Rubber bumpers (4)
13) System ground point
14) Rubber edge moulding

KIM-1 ASSEMBLY

A COUPLE OF BINARY MATH SUBROUTINES FROM: H.T. Gordon, College of Agriculture, Univ of Calif Berkely, California 94720

Program MULTIA (second, revised version). Does binary multiplication of two 8-bit numbers that have been stored (before the JSR to MULTIA) in 00E3 and 00E4 and are destroyed by the operation of the subroutine. The hi 8 bits of the product are stored in 00E0 and the low 8 bits in 00E1; the subroutine initially zeroes these locations, and also 00E2. Operations uses LSRs on the multiplier in 00E4 to move up to 8 bits in sequence into the carry flag. If the carry is set, the multiplicand (in 00E2 and 00E3) is double-precision added to the product locations. If bits remain in the multiplier (00E4 not zero), the multiplicand is shifted left in the 16 bits of 00E2-00E3; otherwise the subroutine exits. Program length 36 bytes. Maximum product (FF X FF) is FE01 or decimal 65025, with execution time about 380 microseconds. Time declines to 240 microseconds for 80 X 80, 160 microseconds for 10 X 10, 70 microseconds for 01 X 01, 40 microseconds for 00 X 00.

NOTE: This subroutine assumes that the processor is in the binary (not the decimal mode)! It should not be necessary for subroutines to protect themselves (by a CLD) from this problem!

```
000A  A9  00     (zeroes locations 00E0 to 00E2)
      85  E2
      85  E1
      85  E0

0012  46  E4     (LSR 00E4, lowest bit into carry)
0014  90  0D     (if carry clear, skip the addition)(go to 0023)

0016  18         (CLC starts double-precision add)
      A5  E1
      65  E3     (running totals stored in 00E0-00E1)
      85  E1
001D  A5  E0
      65  E2
      85  E0

0023  A5  E4     (LDA of 00E4, zero flag set if zero)
      F0  06     (exit to 002D if zero)

0027  06  E3     (ASL shifts highest bit of 00E3 into carry,
      26  E2      ROL shifts carry into lowest bit of 00E2)
002B  90  E5     (carry is always clear, so back to 0012)

002D  60         (RTS exit)
```

SUBROUTINE DIVIDA. This software gives the quotient, to 16-bit or better precision, from division of any hex number from 0001 to FFFF by any hex number from 01 to FF. It uses the 10 lowest locations in zero-page. The quotient appears in the lowest 5, with a fixed decimal implied between 01 and 02. The range of quotients is from $ 0000.010101 (from division of 0001/FF) to $ FFFF.000000 (from division of FFFF/01). Quotient locations are initially zeroed by a JSR to SUBROUTINE ZEROER, which must also be in memory and is coded separately for use in other programs. Before the JSR DIVIDA, 4 locations must be filled by the calling program. The dividend high byte is set in 06, the low byte in 07, and the divisor in 08. The "precision byte", with a value from 01 to 05, is set in location 09; it is not altered by the program, but the other 3 bytes usually are. The purpose of the precision byte is to allow the user to control the number of quotient locations to be calculated by DIVIDA. A value of 01 causes exit after the proper quotient value in location 00 (which may be 00) has been calculated. A value of 02 limits the calculation to quotient locations 00 and 01, and gives "integer arithmetic". A value of 03 allows only one location to the right of the implied decimal, etc.. The chief use is to shorten the execution time, which can approach 2000 microseconds at a precision of 05. However, DIVIDA always exits when the calculated remainder is zero, since calculation of higher-precision locations is then unnecessary. No "rounding-off" operations are included. E.g., the quotient of FEFE/FF is 00FF.FD0000 at a precision of 03, although it should be 00FF.FE since the quotient is 00FF.FDFDFD at a precision of 05.

DIVIDA exits in less than 150 microseconds if the dividend is 0000. It provides no protection against a divisor of 00, so the calling program should guard against this! ~~ag~~ A guard could be inserted in DIVIDA, but I feel it is better for the calling program (to decide) what should be done if such an error occurs.

Operation of DIVIDA involves addition of a shifting single-bit "Bit-Byte" in location 05, to the quotient location controlled by the X-register, whenever a positive remainder is obtained. The X-register is not protected by DIVIDA, so it is better to use Y-indexed loops in the calling program (that otherwise will have to store and restore the X value). The final remainder is in location 06 when DIVIDA exits. The divisor value is not altered if it is $ 80 or more; otherwise it is left-shifted by DIVIDA.

DIVIDA is very long (70 bytes, or 78 if one includes ZEROER; if the zeroing operation were made an integral part of DIVIDA the length would be 74 bytes and execution a shade faster). It is also slow compared to hardware arithmetic, but relatively inexpensive. It is meant to handle data, that are never very precise, and not the kind of complex math for which calculators are designed. Since the ROR instruction is not used, it will run in any 6502 system.

Much of the length of DIVIDAis caused by special logic designed to reduce the execution time---a deliberate trade-off of more program bytes for a lower average time, that has the effect of prolonging the time of divisions where no early exit is possible.

Coding for SUBROUTINE ZEROER.

```
0200   A9  00      (LDA # 00)
       95  FF      (STA  zaro-page, X)
0204   CA          (DEX)
       D0  FB      (BNE, if ≠ 0 back to 0202)
0207   60          (RTS)
```

Coding for SUBROUTINE DIVIDA. (Note that 3 locations are unused between the end of ZEROER and the start of DIVIDA. This is to allow users (if the subroutines are in RAM) to insert 3 instructions following the LDA divisor instruction at 0213. If the divisor is 00, DIVIDA is ~~[illegible]~~ wrong. ~~[illegible]~~. The instructions D0 01 00 substitute for this a BREAK to 1C00. If something more complex is needed, the 3 instructions can be a JMP or JSR to a longer sequence of instructions.)

```
020B   A2  06      (LDX # 06)
       20  00 02   (JSR ZEROER, to zero 0000 to 0005)

0210   38          (SEC)
       26  05      (ROL sets Bit-Byte to 01 and clears carry)
0213   A5  08      (LDA divisor byte)
       30  05      (BMI, if Bit 7 = 1, skip to 021C)
0217   26  05      (ROL Bit-Byte)
       0A          (ASL, left-shift divisor in accumulator)
021A   D0  F9      (BNE, if ≠ 0, back to BMI at 0215)
       85  08      (STA bit-pattern 1XXX XXXX into divisor loc.)

021E   A5  06      (LDA dividend-hi)
       B0  0F      (BCS, if carry set go to subtraction at 0231)
0222   D0  09      (BNE, if ≠ 0, go to CMP at 022D)
       A5  07      (LDA dividend-lo)
0226   F0  28      (BEQ, dividend = 0 so exit to 0250)
       85  06      (STA dividend-lo into dividend-hi location)
022A   86  07      (STX zeroes dividend-lo)
       E8          (INX to shift to next higher quotient loc.)

022D   C5  08      (CMP dividend-hi with divisor)
       90  0B      (BCC, divisor too large, bypass to 023C)
0231   E5  08      (SBC, subtract divisor from dividend-hi)
       85  06      (STA remainder into dividend-hi)
0235   18          (CLC for addition)
       B5  00      (LDA zero-page,X the proper quotient byte)
0238   65  05      (ADC the Bit-Byte)
       95  00      (STA zero-page,X back into quotient loc.)
```

P 17

(completio[illegible] coding for DIVIDA)

```
023C  46  05     (LSR the Bit-Byte)
      D0  09     (BNE, if ≠ 0, bypass resetting)
0240  E8         (INX to shift to next higher quotient loc.)
      E4  09     (CPX to precision-byte)
0243  F0  0B     (BEQ, if equal exit to 0250)
      A9  80     (LDA # 80 to reset)
0247  85  05     (STA into 05 resets Bit-Byte)

0249  06  07     (ASL dividend-lo starts dividend left-shift)
      26  06     (ROL dividend-hi completes the shift)
024D  4C  1E 02  (JMP to 021E for next test sequence)

0250  60         (RTS)
```

Execution time depends both on the number of quotient locations to be filled and on the number of 1-bits to be inserted. Thus, FFFF/01 runs slowly because it requires insertion of 16 1-bits into 2 locations. The "hi/lo exchange" operation at 0228 speeds up many operations with a dividend of 00XX. In general, higher speed will require sacrificing precision, and a precision-byte of 04 will be adequate. My reason for limiting the dividend to 16 bits and divisor to 8 bits was that data more precise than 1 part in 256 will be rare, so that most data will be single-byte, and data sets with more than 256 items will be uncommon. Calculation of the average of 255 one-byte data items is within the capacity of DIVIDA. When there are more, they can be divided into subsets of 255 or fewer, the averages for all subsets added, and the average of the set of subsets calculated. We are now in the time range of seconds! With more bits, it would be minutes. People who need arithmetic speed had better get a 16-bit microprocessor (or better still, shell out for hardware multiply-divide).

Those who want integer arithmetic operations will do better using a dividend of type XX00 and precision-byte of 01. However, similar effects can usually be obtained more quickly and easily by other logic, not division. The number of possible ways of doing division is incredibly large, but I will be surprised if an operation like that of DIVIDA can be done with many fewer bytes or much higher speed, although using the ROR instruction might help.

HERE'S A RELATIVE BRANCH CALCULATOR ETC. FROM: Ted Beach K4MKX, 5112 Williamsburg Blvd., Arlington, Va. 22207

I am enclosing for your perusal two programs I find to be very helpful in writing assembly lanquage programs on KIM.

Also, may I say that the Users Notes has already proven worth the money - even after only two issues. Mr. Butterfield is fantastic, and I really dig his SUPERTAPE - worth the money all by itself! You can add the Norelco 150 Carrycorder to the list of "always works" machines. Mine is about 15 years old and I don't think they are imported any longer, but it is a real good machine.

In addition to the two programs, I have a simple hardware modification which allows the KIM to be started in the monitor mode - a power on reset function which is very easy to add. I built the whole thing on a small piece of perf-board which dangles from the expansion connector.

One other hint I have found helpful when loading multiple section programs (like WUMPUS) is to assign the same ID number, and before loading the first section into KIM, make the following entries:

00EF 73
00F0 18

This is the program counter location, and after getting the first section of the program into KIM, leave the recorder running and punch PC, GO real fast and you're ready for the next section. Also, if you should happen to drop a bit and get an error exit (FFFF 1C), pressing PC gets you back to the tape-read program in a hurry without having to key in 1873.

An there you have it, Eric. Do keep up the good work and let me know if there is anything I can do to help you with the User Notes. I am at present teaching KIM to talk BAUDOT so I can use a cheap 5-level printer soon. Will keep you informed. It looks like a one-page program right now. Next comes Morse, etc., followed by a home-brew ASCII keyboard and more memory.

Branch Calculator - Ted Beach - K4MKX

```
0200  D8           ENTER   CLD
0201  38           COMPUT  SEC
0202  A5 FA                LDA  POINTL
0204  E5 00                SBC  SAV
0206  85 F9                STA  INH
0208  A5 FB                LDA  POINTH
020A  E5 01                SBC  SAV+1
020C  30 0E                BMI  NEG
020E  C9 00                CMP  #0
0210  D0 14                BNE  ERROR
0212  A5 F9        HIOK    LDA  INH
0214  C9 80                CMP  #$80
0216  10 0E                BPL  ERROR
0218  A9 00        OUT     LDA  #0
021A  F0 0C                BEQ  DISP
021C  C9 FF        NEG     CMP  #$FF
021E  D0 06                BNE  ERROR
0220  A5 F9                LDA  INH
0222  C9 80                CMP  #$80
0224  10 F2                BPL  OUT
0226  A9 FF        ERROR   LDA  #$FF
0228  85 FA        DISP    STA  POINTL
022A  85 FB                STA  POINTH

022C  20 1F 1F     BEGIN   JSR  SCANDS
022F  D0 FB                BNE  BEGIN
0231  A9 01        BEGIN1  LDA  #$01
0233  2C 40 17             BIT  SAD
0236  F0 F4                BEQ  BEGIN
0238  20 1F 1F             JSR  SCANDS
023B  F0 F4                BEQ  BEGIN1
023D  20 1F 1F             JSR  SCANDS
0240  F0 EF                BEQ  BEGIN1
0242  20 6A 1F             JSR  GETKEY
0245  C9 15                CMP  #$15
0247  10 E3                BPL  BEGIN
0249  C9 12                CMP  #$12
024B  F0 18                BEQ  SAVE
024D  C9 14                CMP  #$14
024F  F0 B0                BEQ  COMPUT
0251  C9 10                CMP  #$10
0253  10 D7                BPL  BEGIN
0255  0A           DATA    ASL  A
0256  0A                   ASL  A
0257  0A                   ASL  A
0258  0A                   ASL  A
0259  A2 04                LDX  #$04
025B  0A           DATA1   ASL  A
025C  26 FA                ROL  POINTL
025E  26 FB                ROL  POINTH
0260  CA                   DEX
0261  D0 F8                BNE  DATA1
0263  F0 C7                BEQ  BEGIN

0265  A2 01        SAVE    LDX  #$01
0267  B5 FA        SAVE1   LDA  POINTL,X
0269  95 00                STA  SAV,X
026B  CA                   DEX
026C  10 F9                BPL  SAVE1
026E  A2 01                LDX  #$01
0270  F6 00        ADD2    INC  SAV
0272  D0 02                BNE  NEXT
0274  F6 01                INC  SAV+1
0276  CA           NEXT    DEX
0277  10 F7                BPL  ADD2
0279  30 B1                BMI  BEGIN
027B                            (LAST+1)
```

Two page zero locations used: SAV = 0000
SAV+1 = 0001

You can enter the program at any point, however 022C is the best place.

To use the program, enter the address of the branch instruction via the keyboard and press + for enter data. Next enter the destination address of the branch via the keyboard and press PC for perform calculation. The display will show 0000 XX, where XX is the offset value to enter in your program. If you see FFFF XX, the branch is out of range.

The addresses to use in the calculation are the actual line addresses of the instructions, not the actual location of the start of the branch.

For example, to compute the offset for the branch instruction at address 0224 in the program, enter 0 2 2 4 +. The address of the destination (OUT) is entered next : 0 2 1 8 PC and you will see displayed: 0000 F2.

If you make a mistake entering an address, just re-enter it, like you do when using the KIM-I monitor program.

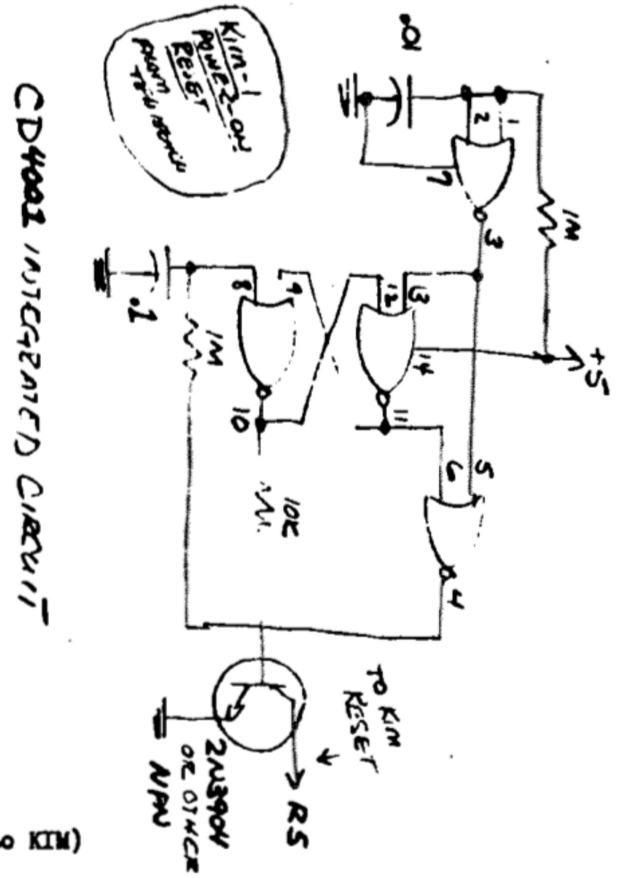
The program is completely re-locatable as there are no jumps other than to KIM subroutines.

(The rather inefficient coding at SAVE happened when I was fooling around with indexed addressing and could be simplified to: LDA POINTL, STA SAV, LDA POINTH, STA SAV+1 - [illegible] bytes instead of [illegible]! Also, the technique used at OUT - 0218 - is handy for a two-byte jump or a branch-always instruction like the 6800 type. Do something to the accumulator, knowing which flags will be affected, and then follow with a branch - real handy on occasions.)

Program P 'er Ted Beach - KIMKX

```
1780  = SAH
1781  = SAL
1782  = EAL+1
1783  = NUM

1784  D8                    CLD
1785  38                    SEC
1786  AD 82 17              LDA   EAL+1
1789  ED 81 17              SBC   SAL
178C  AA                    TAX
178D  AD 80 17              LDA   SAH
1790  85 FB                 STA   POINTH
1792  8D AE 17              STA   17AE
1795  8D B1 17              STA   17B1
1798  AD 81 17              LDA   SAL
179B  85 FA                 STA   POINTL
179D  8D AD 17              STA   17AD
17A0  18                    CLC
17A1  6D 83 17              ADC   NUM
17A4  8D B0 17              STA   17B0
17A7  90 03                 BCC   DOIT
17A9  EE B1 17              INC   17B1
17AC  BD 00 00   DOIT       LDA   SA,X
17AF  9D 00 00              STA   SA+NUM,X
17B2  CA                    DEX
17B3  E0 FF                 CPX   #$FF
17B5  D0 F5                 BNE   DOIT
17B7  4C 4F 1C              JMP   START(return to KIM)
17BA                              (Last+1)
```



This program allows one to move a group of data up in memory so as to be able to insert extra bytes (1 to 255) in a program. The address of the first byte to be moved is entered in 1780, 1781. The lower address of the last byte+1 in the program is put in 1782. The number of bytes you wish to insert in the program is put in 1783. Press + and GO and it's done. The program exits to KIM with the required starting address displayed so you can write in your data immediately by pressing DA.

As an example, I wanted to add two bytes to the Moon Lander program so that when in the FUEL mode, the display would show ~~$ F000 XX~~. This Fxxx xx required an ORA #$F0 instruction just after SHOFL (0063). The address of the first byte to be moved is 0065, which goes in 1780 and 1781. The last byte+1 to be moved (lower address only) is C5 which is entered in 1782. I need to add two bytes, so 02 goes in 1783. Press + and GO and you'll see 0065 A6 displayed. Push DA, 09 + F0 and you've done it.

Now you'll have to change any branches which cross the patch by 02, and you'll have to add 02 to all addresses which follow the patch, as well as change the jump addresses... no real problem, and it sure beats re-entering all that data by hand!

From the Editor - Mr. Beach's relative branch calculator is really slick! I'm sure that all of us will be able to save plenty of time. An absolute branch can also be achieved with 3 bytes by CLC BCC etc.

The program patcher is a good starting point for a general purpose "move bytes" routine that could move bytes up or down in memory. It should work with programs of ANY length and may have to cross page boundaries. Be aware, though, that if the move routine is moving bytes to lower addresses, it will have to start pulling bytes down from the bottom end working up to the top end and vice versa for moving bytes upward in memory. Teds program patcher will not cross page boundaries and can only move bytes up in memory.

And some more ideas for utility programs: A couple of articles in the Feb. issue of Kilobaud really got the grey cells into a flurry of activity. The first article "Chasing Those Naughty Bits" (John Molnar) described an interesting "walking-bit" memory test program for a 6800 that could very easily be rewritten for the 6502. John also introduced the readers to the idea of a "SYSTEM EXERCISER" program which would test ALL machine instructions capable of being executed by the CPU. Another idea for the 6500 system. A SYSTEM EXERCISER and memory test program should be included with every system to instill confidence in the new system owner.

Kilobaud #2 also had an article of interest to those of us with termin

The article "Cut programming time with this extraordinary Program" (Mark Borgenson P.104) described a hex loader routine to make loading programs go alot smoother. The Motorola "Mikbug" is similiar to the Kim Monitor in that as you load the memory, via terminal, each memory location is printed out along with the contents of that location. To access the next location, you punch the keyboard again and go through the whole procedure again and again. Borgenson's alternative, (written for 6800 again) allows you to type in as many bytes as will fit on a line (whatever your terminal line length is). When going to a new line, the computer could indicate the next available address to be loaded. Provisions were included to allow backspacing the memory pointer to allow for error corrections.

Converting programs from the 6800 to the 6502 is relatively straight forward once you are familiar with the program operation.

Understanding the 6800 instruction set is again straightforward once you know the 6502.

Several new books came out recently which, although written about the 6800, could be useful to our purposes. I have not read either of them yet and would appreciate a review of these publications by someone who has read them.

"Scelbi "6800" Software Gourmet Guide" ($9.95)
Scelbi Computer Consulting Inc.
1322 Rear Boston Post Road
Milford, Ct. 06460 Tel. 203-874-1573

"6800 Programming For Logic Design" $7.50
#5001 Adam Osborne

Remember the Assembler - Text Editor that was mentioned on page 1 issue #1? Well, after receiving several letters indicating that the assembler/text-editor had not yet been delivered. I decided to find out what the problem was. I talked with David Snow, president of Micro-Software Specialists. He stated that they were running into difficulties getting the assembler together and had notified all who had ordered the package that they could have a refund if they so desired. He also said that they hoped to have the package together shortly and further stated that the package would not be totally compatible with MOS Technology's cross assembler mnemonics and operations (as had been indicated in a spec sheet that I had received).

I would never have mentioned this had I known that it was not yet available, and recommend that you hold off ordering anything from this Company until I can verify that their software is, in fact, available for purchase.

While I am on the subject, I have also received several letters indicating problems with interfacing KIM to OSI memory boards. One possible problem became apparent after a cursory inspection of the OSI application note on the subject. Address lines 10, 11, and 12 are driving a 74145 decoder on the KIM board. OSI recommends the use of 7417 buffers on all address lines. Since a 6502 can only drive 1 TTL load, problems can arise when address lines 10, 11, and 12 attempt to drive 2 TTL loads, as would be the case when driving both the on board 74145 and the 7417. Does anyone have more information about this interface?

One solution to the problem would be to simply replace the on board 74145 decoder with a 74LS145 low power Schottky device and use 8T97 buffers instead of the 7417's. The only problem is trying to find a 74LS145 (can anyone help out?).

Some of you have no doubt received the "uP4" (a newsletter) from Johnson Computer, P.O. Box 523, Medina, Ohio 44256. (Tel. # 216-725-4560). Although primarily an advertising brochure, the "uP4" also contains data on MOS Techs latest chip offerings and other tidbits of information of interest to most 6502 users. If you haven't received it, and want a free copy, write to them. They stock most MOS products (except for the KIM 4 Motherboard and the 6522 VIA - neither of which has been released as of yet). They also stock the MOS 7529-103 Scientific Calculator Chip (available for $20.65, including postage, handling, and the 20 page documentation package) which will be interfaced to KIM in the next issue of the KIM-1 User Notes. I bought my chip from them so I know for sure that they stock them. They also stock Kim-2,3.

Coming Up In The User Notes

The Ultimate Calculator Interface - uses the 7529-103 calc. chip, and interfaces directly to KIM's I/O ports with the addition of 9-750 ohm resistors. No special power requirements (just 5 volts and ground) and no other chips are needed. A basic software driver will also be included. This interface was released as an application note from MOS but the I/O configuration and software had to be modified for KIM-1 operation.

KIM-1 Expansion - A series of articles will be started pertaining to the expansion of KIM into a complete system.
A CHESS CLOCK, REAL-TIME CLOCKS, HARDWARE INTERFACES
Plus more games - music programs, and letters from Users, etc, etc.
(also reprints from the Complimentary July issue of USER NOTES)

On PLEASE

I received complete PLEASE software package (mentioned in issue #2) from MicroCosmos, 210 Daniel Webster Hwy., So. Nashua, N.H. 03060. Even though I couldn't get the cassette tape to load correctly, I was quite impressed with the documentation that was provided. The source and object code listing is very well commented and the interpreter and monitor routines are explained in full detail. PLEASE includes a number of interesting games, puzzles and useful routines (hex to decimal, decimal to hex, among others). Information is presented which ill enable you to use the PLEASE monitor routines for you own programs. I have talked with several people who had no trouble reading the cassette so my case is probably the exception rather than the rule. Robert Tripp, author of PLEASE, has indicated plans for making other software packages available in the future.

On MICROCHESS

Peter Jennings says that his chess playing program is now available and runs on the basic KIM with no additional memory or I/O. Peter goes on to say that for $10.00 he will provide a Players Manual, a Programmers Manual, and a complete commented source listing of the program. He also states that the program is expandable, provided additional memory is added to KIM. For more information contact: MICROCHESS, 1612-43 Thorncliffe Pk. Drive, Toronto, Ontario, Canada M4H 1J4

•••••••••••

If you have interfaced KIM to the Southwest Technical Products GT-6144 Graphic Display (or if you are planning on doing so) Stan Ockers, RR#4, Box 209, Lockport, Ill. 60441, would like to exchange ideas with you. Stan says he has it up and running.

Rene Vega, 4211 Avery, Detroit, Mich. 48208, had added some enhancements to Tiny Basic, plans more, and wants to exchange thoughts with interested people.

•••••••••••

HORSERACE

two offerings from:
Charles K. Eaton
19606 Gary Ave.,
Sunnyvale, Cal
94086

Eight lap horse race and you can be the jockey and whip your horse to go faster. Warning-- whip the horse too much and he probably poops out.

Horse	Track	Whipping button
Prince Charming	top	PC
Colorado Cowboy	middle	C
Irish Rair	bottom	4

Start program at 027F. Race is 8 laps.

```
0270 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX D8
0280 A2 13 BD 7C 03 95 7C CA 10 F8 A9 7F 8D 41 17 A0
0290 00 A2 09 B9 7C 00 84 FC 20 4E 1F C8 C0 06 90 F3
02A0 20 3D 1F A5 8F 30 E3 A2 03 CA 30 DE D6 86 D0 F9
02B0 86 99 A4 99 B6 83 B9 90 03 35 7C EA EA EA EA EA
02C0 95 7C E8 96 83 B9 90 03 49 FF 15 7C 95 7C E0 05
02D0 30 38 D0 06 A5 8F F0 28 D0 30 A2 02 38 B5 83 E9
02E0 06 95 83 CA 10 F6 A2 06 B5 7C 95 76 A9 80 95 7C
02F0 CA D0 F5 EA EA EA EA EA EA EA EA EA EA EA EA EA
0300 C6 8F D0 06 A5 81 09 06 85 81 EA EA EA EA EA EA
0310 B9 89 00 F0 0B 20 68 03 29 3C D0 1B 99 89 00 EA
0320 20 68 03 29 38 85 9A B9 8C 00 30 0B 29 38 c5 9A
0330 B0 05 A9 FF 99 89 00 20 3d 1F A0 FF A6 99 3D 93
0340 03 f0 01 88 98 55 89 85 9A EA EA 20 68 03 38 29
0350 01 65 9A 18 A6 99 75 8C EA EA EA EA EA EA EA EA
0360 95 8c 95 86 4c a9 02 xx 38 A5 92 65 95 65 96 85
0370 91 A2 04 B5 91 95 92 CA 10 F9 60 XX 80 80 80 80
0380 80 80 80 FF FF FF 80 80 80 00 00 00 80 80 80 08
0390 FE BF F7 01 02 04
```

Thanks to Jim Butterfield for the original idea, my wife for the whipping suggestion, and Jim Butterfield, again, for the dispaay routine and random number generator.

NOTE ON MOONLANDER

Terrific program, but we were wondering why it sometimes slips into feul mode. it turns out this is from using the GETKEY subroutine in decimal mode. This routine sets the accumulator equal to 15 (Hex) if it finds no key pushed. But that's the decimal code for 'F', so under the condition that it braches into GETKEY because there is a key depressed, and the key is lifted before execution of the subroutine, MOONLANDER interprets the 15 default value as a depressed 'F' key. Easiest solution--change step 0092 to 14 and now 'E' (Energy) will be the button for fuel mode.

John L. Aker
701 S. Highland
Chanute, Kansas 66720

David G. Alexander
2628 Ribier Way
Rancho Cordova, Ca. 95670

H. M. Allen
24674 Cushing
East Detroit, Mi. 48021

Paul S. Allen
507 Hill Street #5
Santa Monica, Ca. 90405

Dr. Ronald J. Allen
Kapteyn Laboratorium
Hoogbouw W.S.N.
Postbus 800
Groningen Nederland
Holland

Richard F. Amon
3782 Montego Drive
Huntington Beach, Ca. 92649

Carl Andeen
18720 Monterey Ave.
Cleveland, Ohio 44119

V. E. Anthony
15911 Willbriar
Mo. City, Tx. 77459

Richard J. Archer
6229 S. W. Erickson
Beaverton, Ore. 97005

Charles Arnold
151 Prescott Ave.
Staten Island, N.Y. 10306

Michael S. Askins
2530 Hillegass #201
Berkeley, Ca. 94704

Arthur J. Aspere
352 Urban Lane
Cecil, N.J. 08094

Kenneth Aupperle
24 Arrowwood Lane
Melville, New York 11746

Ronald G. Axium
10330 S. Walden Pky.
Chicago, Ill. 60643

David A. Baker
614 Lynwood Ct.
Richland, Wash. 99352

Dennis Baker
#1201
11700 Old Columbia Pike
Silver Spring, Md. 20904

T. A. Balasubramaniam Ph.D.
752 Metropolitan Avenue #10
Hyde Park, Mass. 02136

Henry Ball
1066 E. Delaware
Brubank, Ca. 91504

Charles B. Balser
14 Overbrook Circle
New Hartford, N.Y. 13413

E. E. Barnes
Star Route
Parkesburg, Pa. 19365

Joseph B. Bartone, Pres.
All Purpose Comm. Corp.
168 Fifth Avenue
New York, N.Y. 10010

H. Bausa
Transidyne General Corp.
903 Airport Drive
Ann Arbor, Mich. 48104

John Baxter
228 Manor Drive
Richboro, Pa. 18954

Edward B. Beach
5112 Williamsburg Blvd.
Arlington, Va. 22207

Elmer T. Beachley
5601 Penn Avenue
Pittsburgh, Pa. 15206

Bibliotheek Der Rijksuniversiteit
afd. Kapteyn Lab.
O. Kijk in 't Jatstraat 5
Postbus 559
Groningen - Nederland

Conrad C. Bjerke
512 Camino de Encanto
Redondo Beach, Ca. 90277

Kenneth Blackledge
1103 K. Avenue
Nevada, Iowa 50201

Lowell E. Blevins
3225 Highland
Muskegon Hgts., Mi. 49444

A. M. Blum
10 Killington
Chappaqua, N.Y. 10514

Truman Boerkoel
2050 Brookridge Drive
Dayton, Ohio 45431

I. R. Bogue
Rogers Corporation
Willimantic Div.
Engineered Products Group
P.O. Box 126
Willimantic, Conn. 06226

Blair E. Bona
164 Janine Dr.
La Habra Heights, Ca. 90631

Tom Bonfield
7411 Carta Valley Dr.
Dallas, Tx. 75248

Donald L. Box
1250 White Oak Dr.
Cookeville, Tenn. 38501

Walter adley
1315 Nasa Rd. 1, No. 152
Houston, Tx. 77058

David C. Braught
Illinois Wesleyan Univ.
Bloomington, Ill. 61701

James Brick
820 Sweetbey Dr.
Sunnyvale, Ca. 94086

Larry Briggs
16719 Superior
Sepulveda, Ca. 91907

Stan Brindle
Box 85, MSU
Lake Charles, La. 70601

James D. Bruce
6534 Foyle Way
San Diego, Ca. 92117

James S. Bryan
Compumetric Corporation
3206 Hoffman St.
Harrisburg, Pa. 17110

Samuel W. Burridge
30 Everson Street #305
Denver, Co. 80218

C.A.C.H.E.
P.O. Box 36
Vernon Hills, Ill. 60061

D. Kingston Cable
4817 E. Sheila Street
Los Angeles, Ca. 90040

James L. Calvert M.D.
900 Kiely Blvd.
Santa Clara, Ca. 95051

Arvind Caprihan
Rua UCA 551 / #201
Jardim Guanabara
Ilha Do Governador
Rio De Janeiro, Brasil

Wilson Caselli
Design Engineer
Fycon-Division of CSF, Inc.
3609 Tryclan Dr.
Charlotte, N C. 28210

Celanese Corp.
1211 Avenue of the Americas
N.Y., N.Y. 10036

John W. Clark
Niagara College
Woodlawn Road
Welland, Ontario L3B 5S2

Coburn Optical Indus., Inc.
1701 South Cherokee
Muskogee, Okla. 74401

Philip H. Cochran
Offshore Navigation, Inc.
P.O. Box 23504
Harahan, La. 70183

John Conti
Audio Scien(, Inc.
1165 East 15th Street
Brooklyn, N.Y. 11230

Donald C. Courtney
1862 Hopkins Dr.
Wixom, Mich. 48096

John Craig, Editor
Kilobyte
RFD Box 100D
Lompoc, Ca. 93436

Chris Crawford
1766 26th Ave.
Columbus, Neb. 68601

F. L. Crawford, Jr.
2132 Carolina Drive NE
Cedar Rapids, Iowa 52402

Robert H. Cushman
22 Orchard Farm Rd.
Port Washington, N.Y. 11050

Robert Dahlstrom
Microwave Branch 150
Harry Diamond Laboratories
2800 Powder Mill Road
Adelphi, Md. 20783

Paul K. Dano
517 East Cliff Drive
Euless, Tx.

Data Systems Associates
313 North First St.
lompoc, Ca. 93436

Steve Davidson
170 Greenview Circle
Ventura, Ca. 93003

James R. Davis
Attorney At Law
202 Title Building
Birmingham, Ala. 35203

Katherine Davis
130 W. 16th St.
New York, N.Y. 10011

J. R. Davis
Research & Development
1310 Beulah Road
Churchill Boro
Pittsburgh, Pa. 15235

Richard Deal
6957 Saroni
Oakland, Ca. 94611

Ralph Deane
Box 38
Littlefort, B.C.
Canada VOE 2CO

Bruce J. DeGrasse
3311 Brookhaven Club Drive
Farmers Branch, Tx. 75234

Arthur Deland
3817 Gay Road
Erie, Pa. 16510

Robert Denison
Tri-State Bus Company
RD5 Teeter Rd.
Ithaca, N.Y. 14850

Allen G. Dennison
Instrumentation Laboratory Inc.
Jonspin Road
Wilmington, Mass. 01887

Wm. R. Dial
438 Roslyn Avenue
Akron, Ohio 44320

John L. Dickinson
3 Thendara Lane
Poughkeepsie, N.Y. 12603

Harry DiLisio
326 Sefler St.
McKees Rocks, Pa. 15136

Charles E. Doenlen
4421 Devereaux St.
Philadelphia, Pa. 19135

Michael Dolan
103 Alexandria Drive
Oxon Hill, Md. 20021

J. O. Donnell
18 Blake Ave.
Rockledge, Pa. 19111

Dow Badische Company
Drawer 3025
Anderson, S. C. 29621

John Doyle
25 Woodridge Crescent
Ottawa, Ontario Canada K2B 7T4

Charles A. Drace
762 Barron Ave.
Palo Alto, Ca. 94306

Pete Dudley
16359 Shady View Ln.
Los Gatos, Ca. 95030

Bob Duke
13526 Pyramid Dr.
Dallas, Tx. 75234

Michael S. Duncan
1468 Lakeshore Cr. N. W.
Gainesville, Ga. 30501

Nelson Dunn
122 East 27th Street
New York, N.Y. 10016

J. S. Dworzan
368 S. Carolet Lane
Orange, Ca. 92669

Charles K. Eaton
19606 Gary Avenue
Sunnyvale, Ca. 94086

John Eaton
435 West Padre #W10
Santa Barbara, Ca. 93105

Lewis Edwards, Jr.
1451 Hamilton Ave.
Trenton, N.J. 08629

F. H. Edwards - Assoc. Prof.
Univ. of Massachusetts
Amherst, Mass. 01002

Lionel M. Edwards
Harford Community College
401 Thomas Run Road
Bel Air, Maryland 21014

Louis J. Edwards
3462 South 3125 East
Salt Lake City, Utah 84101

Mark W. Edwards
179-7 Evergreen Terr.
Carbondale, Ill. 62901

Electro-Diagnostic Instrum.
819 South Main Street
Burbank, Ca. 91506

Jacob Elias
Box 488
Niverville, Manitoba ROA 1EO

Robin Ellwood
18506 Dylan Street
Northridge, Ca. 91324

Richard B. Emerson
158 E. Butler Avenue
Ambler, Pa. 19002

Engineered Systems, Inc.
1841 East 3rd. St.
Tempe, Arizona 85281

Kenneth W. Ensele
1337 Foster Road
Napa, Ca. 94558

R. G. Eschenbach
1833 Edlewild
Burlington, Ky. 41005

Nicola Evangelista
230 Brook Avenue
Bronx, N.Y. 10454

E. Eyler
Box 2211
Ann Arbor, Mich. 48106

Randy Fallgatter
7910 RioVista Drive
Goleta, Ca. 93017

Clyde Farris
9618 Richeon
Downey, Ca. 90240

Alan Feldman
4409 W. Lunt
Lincolnwood, Ill. 60646

R. Alan Feltner
Corco, Inc.
6950 Worthington - Galena Rd.
Worthington, Ohio 43085

M. B. Fermmim
AKE Sjodin
Bangata 43A
S-72228 Vasteras Sweden

R. Ferriaulo
Afsouth PO
FPO NY, N.Y. 09524

Dr. F. Fiedler
Institut Fur Klinische Chemie
Der Universitat Munchen
8 Munchen 15
Nussbaumstrasse 20 - Ruf 539911

R. Wayne Fields, Ph.D.
Biophysics Laboratory
University of Oregon
611 S. W. Campus Drive
Portland, Oregon 97201

Frederic N. Firestone
806 N. School St.
Normal, Ill. 61761

Mike Firth
104 N. St. Mary
Dallas, Tx. 75214

Stephen H. Fischer
1275 Picasso Drive
Sunnyvale, Ca. 94087

Howard Fisher
Box 495
Mary Esther, Fla. 32569

Christopher J. Flynn
2601 Claxton Drive
Herndon, Va. 22070

Bern Fox
Environmental Res. Inst.
of Michigan
P.O. Box 618
Ann Arbor, Mich. 48107

Mervin E. Frank
Box 11747
Santa Ana, Ca. 92711

Harvey S. Frey, M.D.,Ph.D.
552 12 St.
Santa Monica, Ca. 90402

Eliot I. Friedman
1175 Wendy Rd.
Ann Arbor, Mich. 48103

David E. Fulton
Versa-Chem Corp.
121 Meadow St.
Hackensack, N.J. 07601

Dr. S. Y. Fung
University of Calif.
Physics Dept.
Riverside, Ca. 92502

Dan Fylstra
Hamilton C-23
Harvard Business School
Soldiers Field
Boston, Mass. 02163

Eugene R. Garapic
14231 Thompson Blvd.
Brook Park, Ohio 44142

Dan Gardner
11825 Beach Blvd.
Stanton, Ca. 90680

Jerry Garratt
6411 E. 9th Avenue
Spokane, Wash. 99206

Dr. Martin A. Garstens
913 Buckingham Drive
Silver Spring, Md. 20901

David A. Gauger
% Littlefuse Inc.
800 E. Northwest Hwy.
Des Plaines, Ill. 60016

Dr. James W. Gault
4916 Liles Rd.
Raleigh, N.C. 27606

Gedee Associates
Gopal Bagh
P.O. Box 3755
315 Avanashi Road
Coimbatore 641018 India

John C. Geiger
5643 S. 39 Street
Milwaukee, Wis. 53221

Morris Gisser
609 Winona Court
Silver Spring, Maryland 20902

R. Gladstone
Gladstone Electronic Supply
1736 Avenue Road
Toronto, Ontario M5M 3Y7

Mr. Indulis Gleske
Physics Department
University of New Hampshire
Durham, New Hampshire 03824

Philip Goembel
1413 John Street
Baltimore, Md. 21217

Markus P. Goenner
Uetlibergstr 107
8045 Zurich Switzerland

Ray Golden
Controlomation
P.O. Box 1465
Ann Arbor, Mi. 48106

John C. Goodman
Polaroid Corporation
565 Technology Square -7C
Cambridge, Mass. 02139

Sperry H. Goodman
2220 Minor Ave. E.
Seattle, Wa. 98102

Harold T. Gordon
641 Paloma Avenue
Oakland, Ca. 94610

Richard Greene
163 Eastern Parkway
Brooklyn, N.Y. 11238

A. J. Greer
4216 GrapeVi. .rive
Napa, Ca. 94558

C. A. Gregory, Jr.
106 Tempsford Land
Richmond, Va. 23226

Wilfred J. Gregson, II
4104 Wadsworth Ct., No. 202
Annandale, Va. 22003

Katiya Greigarn
P.O. Box 834
Rolla, Mo. 65401

Jerome D. Groskind, Atty.
6 Beacon Street
Boston, Ma. 02108

James Grover
1221 Carriage Hill Drive
Athens, Ohio 45701

M. A. Grumboski
2831 Ewald Circle
Detroit, Mi. 48238

Robert D. Gustafson
609 W. Stratford Pl.
Chicago, Ill. 60657

Donald J. Hall
1629 E. Cooke Rd.
Columbus, Ohio 43224

Christian Haller
2916 Seymour Drive
Dallas, Tx. 75229

Jim Halverson
9956 Johnnycake B7
Painesville, Ohio 44077

John S. Hamlet
3482 North Summit Ave.
Milwaukee, Wi. 53211

James E. Hamlin
3896 Bancroft Road
Bellingham, Washington 98225

O. F. Hamm
4751 Louisiana Ave.
St. Louis, Mo. 63111

Thomas A. Hanson
6935 York Rd. #207
Parma Hgts., Ohio 44130

William Hapgood
Thermal Systems Engineer
Raytheon Company
Foundry Avenue
Waltham, Mass. 02154

Glyn Harding
Applied Technology
645 Almanor Ave.
Sunnyvale, Ca. 94086

Paul T. Harper
1435 Richardson Ave.
Los Altos, Ca. 94022

George P. Harris
% Home C(er Company
P.O. Box
University Station
Charlottesville, Va. 22903

Earl J. Hart
1477 Braddock Lane
Penn Wayne, Pa. 19151

Maurice I. Hart
Chemistry Department
University of Scranton
Scranton, Pa. 18510

John S. Hasling
1923 Sussex Lane
Colorado Springs, Col. 80909

F. A. Hatfield
Computar Data Systems, Inc.
1372 Grandview Avenue
Columbus, Ohio 43212

Tom Haworth
31098 Rivers Edge
Birmingham, Mich. 48010

Arnold E. Hayes
8645 Newton Place
Manassas, Va. 22110

HDE, Inc.
Box 120
Allamuchy, N.J. 07820

Joel Heckman
234 Tennyson Terrace
Williamsville, N.Y. 14221

Mr. Harvey Heinz
9730 Townline Diversion
Surrey, B. C. V3V 2T2

Carl A. Helber
Helber Research
7406 E. Butherus Drive
Suite F
Scottsdale, Arizona 85254

Philip M. Henderson
4040 Clardon Drive
Williamsville, N.Y. 14221

Curtis K. Henrickson
1 N 555 Forest Avenue
Glen Ellyn, Ill. 60137

Christopher Hentschel
9 West Street
Kingston, Ma. 02364

Robert J. Herbold
Computer Center
Gallaudet College
Kendall Green, Wa. D.C. 20002

Dr. Harvey B. Herman
Department of Chemistry
Univ. of N.C. At Greensboro
Greensboro, N. C. 27412

Carlos Herrin
4118 Tareyton Drive
Bellbrook, Ohio 45305

Robert F. Herrman, P.E.
702 Bowling Green
Moorestown, New Jersey 08057

James G. Herrmann
3758 N. Troy St.
Chicago, Ill. 60618

Richard Ho
Adm 275, Inst. Research
University of Nebraska
Omaha, Ne. 68101

Gregory F. Hogg
0627 O'Shaughnesay; UPI & SU
Blacksburg, Va. 24061

Philip B. Hollander, Ph.D.
The Ohio State University
College of Medicine
Department of Pharmacology
1645 Neil Avenue
Columbus, Ohio 43210

William Holmgren
2603 Meade Circle
Colorado Springs, Col. 80907

Jim Hoo
120-10th Ave. East
Apt. A
Seatlle, Wa. 98102

Ray Hoobler
789 West End Ave. #4D
New York, N.Y. 10025

Alan Hoover
7073 Brookview Road
Hollins, Va. 24019

Jeffrey A. Hoover
1420 Wilder Avenue #34
Honolulu, Hawaii 96822

Wayne C. Horton
2330 W. Shaw Butte Dr.
Phoenix, Arizona 85029

Donald E. Houghton
6590 Buckland Avenue
West Bloomfield, Mich. 48033

Thomas W. Hubbell
533 Wintergreen Dr.
Victor, N.Y. 14564

George B. Hudson
1425 Peacock Lane
Saint Louis, Mo. 63144

John Humphrey
WUOB
Ohio University
Athens, Ohio 45701

John J. Hus
2762 Wisner
Waterford, Mich. 48095

Roy E. Hutchinson
1052 Woodhill
Newark, N.Y. 14513

Henry Parrondo
Ideal Corporation
1000 Pennsylvania Ave.
Brooklyn, N.Y. 11207

Jeff Imig
212 Prospect Apt. U196
Bloomington, Ill. 61701

J. Innis
718 Juniper Dr.
Newport News, Va. 23601

Donald L. Ismert
41 Legion Drive
Kenmore, N.Y. 14217

John A. Jaecker
2352 Clyde Ter.
Homewood, Ill. 60430

Bill Jellerson
7100 138 Place N.E.
Redmond, Wash. 98052

Gerald C. Jenkins
774 Twin Branch Dri.
Birmingham, Ala. 35226

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, Ca. 91103

Brion Johnson
6725 Abrego #16
Isla Vista, Ca. 93017

Norman F. Johnson
3400 Rocky Point Rd.
Bremerton, Wash. 98310

Phil Johnson
McShane, Inc.
P.O. Box 523
Medina, Oh. 44256

Richard Kashdan
206 Acadia Street
San Francisco, Ca. 94131

Marko Katic
777A St. Clair Ave. W.
Toronto, Ont. Canada
M6C 1B7

M. J. Keady
2106 Laurel Hill
Kingwood, Tx. 77339

Thomas Keenan
807 Oxford Blvd.
Steubenville, Oh. 43952

Edward W. Keil
Rt. 2 Box 473
Manchester, Missouri 63011

Don Keller
Cash Register Sales, Inc.
124 N. Braod Ave.
New Orleans, La. 70119

Dick Kelly
P.O. Box 1635
Palo Alto, Ca. 94302

Wallace Kendall
9002 Dunloggin Rd.
Ellicott City, Md. 21043

Michael L. Kern
70 West Camano Drive
Camano Island, Wash. 98292

FROM THE FACTORY

~~FROM THE FACTORY~~: Problems with <u>some</u> 6502 chips.

I have been informed of two possible problem areas with the 6502 CPU. Some of the early CPU's didn't set the zero flag correctly after register to register transfers.(TAY, TAX,TYA etc) This would make it impossible to run Tiny Basic. All 6502 CPUs could have problems setting the zero flag after decimal addition when the answer equalled zero. Test your CPU with a simple program to see if you have these problems. The decimal addition problem can be gotten around by an "OR" immediate with "00" after the operation which would then set the zero flag correctly. The only method of solving the register transfer not setting zero flag would be to install a new CPU. (You would then pick up the "ROR" rotate right instruction in the process).

Copying KIM cassette tapes:

If you've tried copying KIM tapes from cassette to cassette then you already know that the copies will not be read correctly by KIM. The fix? Simply tie a .02 uf capacitor to ground from the junction of R6 and R34. Now make the new master tape. Copies from this new master should be read correctly by KIM.

KIM-2 Memory expansion:

If you wish your KIM-2 (4K) memory module to reside in the 0400-13FF position already decoded by KIM, then read on; Wire - or the KIM K1, K2, K3, K4 decoder outputs together, add a 3.3K pull-up resistor to +5 volts and tie these four lines to the address line 12 input on the KIM-2, set all the on-board DIP switches to the "off" position, and you're all set!

••••••••••

INTRODUCTION TO KIM-1 ARCHITECTURE

J. Butterfield
14 Brooklyn Avenue
Toronto M4M 2X5
Canada

This is intended to be a beginner's guide to the way KIM-1 is put together. It's mostly a hardware description with (hopefully) explanatory notes.

Because every KIM-1 owner has three fat manuals on his system, complete with detailed drawings, I'll try to save space here by referring to these manuals wherever possible.

<u>The Address and Data Busses</u>.

Let's start with page 24 of the KIM-1 User Manual (KIM-1 Block Diagram, Figure 3.1). We're going to concentrate on those two pipes on the left: the Address and Data Busses.

Every microsecond, the 6502 microprocessor sends out an address over the sixteen lines of the address bus. Sixteen lines are enough to address any of the 65,536 memory locations that could be fitted to KIM-1 (you only have 3,328 active addresses in the basic unit). In addition, there are a couple of other lines that accompany this bus: a Read/Write line (R/W) to tell whether the microprocessor wants to read or write memory; and a timing line, ø2 (phi, pronounced fy, two; it's confusing on the diagram because the phi looks like a zero).

The address bus goes to all memories. The idea is that when an address is generated, one memory only (whether RAM, ROM, I/O or Timer) suddenly says, 'that's me!' and connects to the data bus. If the R/W line says, 'read', the memory concerned places its data onto the bus; if 'write', the memory takes the data from the bus (placed there by the microprocessor) and stores it. For every address, only one memory unit responds; the rest stay silent.

This points out a fundamental difference between the address and data busses. The address bus goes one way only: from the processor to the memory. But the data bus information flows both ways.

This calls for a special kind of circuitry to connect to the data bus, called 'tri-state'. Every device on the bus might be (1) sending; (2) receiving; or, (3) ignoring the bus. (That isn't exactly how tri-state is defined, but it's a good way to remember it).

<u>LOOKING FOR PROMS????</u> I just picked up four 82S129 bipolar 256x4 proms to make SUPERTAPE and other much used software a permanent part of KIM. S.D. SALES CO., P.O. BOX 28810, DALLAS, TEXAS, 75228, also has 5204 EPROMS (512x8, 1 usec.) for $7.95, 1702A (1 usec. 256x8) for $6.95, 8T97B's for $1.25, etc. etc. All their parts are guarenteed and they deliver in about a week. Definitely good people! Get their catalog! If you order from this, include 5% for shipping, $0.75 for orders under $10.00, and tax if you're in Texas.

—the editor—